Faster Chosen-Key Distinguishers on Reduced-Round AES

Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean

École Normale Supérieure, 45 Rue d'Ulm, 75005 Paris, France
{Patrick.Derbez,Pierre-Alain.Fouque,Jeremy.Jean}@ens.fr

Abstract. In this paper, we study the AES block cipher in the chosen-key setting. The adversary's goal of this security model is to find triplets (m, m', k) satisfying some properties more efficiently for the AES scheme than generic attacks. It is a restriction of the classical chosen-key model, since as it has been defined originally, differences in the keys are possible. This model is related to the known-key setting, where the adversary receives a key k, and tries to find a pair of messages (m, m') that has some property more efficiently than generic attacks. Both models have been called open-key model in the literature and are interesting for the security of AES-based hash functions.
Here, we show that in the chosen-key setting, attacking seven rounds (resp. eight rounds) of AES-128 can be done in time and memory 2^8 (resp. 2^{24}) while the generic attack would require 2^{64} computations as a variant of the birthday paradox can be used to predict the generic complexity. We have checked our results experimentally and we extend them to distinguisers of AES-256.

Keywords: AES, Open-key Model, Chosen-key Distinguisher, Practical Complexities.

1 Introduction

The Advanced Encryption Standard (AES) [16] is nowadays the subject of many attention since attacks coming from hash function cryptanalysis have put its security into question. Related-key attacks and meet-in-the-middle attacks that begin in the middle of the cipher (also known as splice-and-cut attacks) have been proposed to attack the full number of rounds for each AES versions [1,2,4], while other techniques exist for smaller version [5]. This interesting connection between hash functions and block ciphers shows that any improvement on hash function cryptanalysis can be useful for attacking block ciphers and vice-versa.

In this work, we study another model that has been suggested to study the security of hash functions based on AES components. Knudsen and Rijmen [9] have proposed to consider *known-key* attacks since in the hash function domain, the key is usually known and the goal is to find two input messages that satisfy some interesting relations. In some setting, a part of the key can also be chosen (for instance when salt is added to the hash function) and therefore, cryptanalysts have also consider the model where the key is under the control of the adversary. The latter model has been called *chosen-key* model and both models belong to the *open-key* model. The chosen-key model has been popularized by Biryukov et al. in [2], since a distinguisher in this model has been extended to a related-key attack on the full AES-256 version.

Related Work. Knudsen and Rijmen in [9] have been the firsts to consider known-key distinguishers on AES and Feistel schemes. The main motivations for this model are the following:

– if there is no distinguisher when the key is known, then there will also be no distinguisher when the key is secret,

– if it is possible to find an efficient distinguisher, finding partial collision on the output of the cipher more efficiently than birthday paradox would predict even though the key is known, then the authors would not recommend the use of such cipher,
– finally, such model where the key is known or chosen can be interesting to study the use of cipher in a compression function for a hash function.

In the same work, they present some results on Feistel schemes and on the AES. Following this work, Minier et al. in [14] extend the results on AES on the Rijndael scheme with larger block-size.

In [2], Biryukov et al. have been the firsts to consider the chosen-key distinguisher for the full 256-bit key AES. They show that in time $q \cdot 2^{67}$, it is possible to construct q-multicollision on Davies-Meyer compression function using AES-256, whereas for an ideal cipher, it would require on average $q \cdot 2^{\frac{q-1}{q+1}128}$ time complexity. In these chosen-key distinguishers, the adversary is allowed to put difference also in the key. Later, Nikolic et al. in [15], describe known-key and chosen-key distinguishers on Feistel and Substitution-Permutation Networks (SPN). The notion of chosen-key distinguisher is more general than the model that we use: here, we let the adversary choose the key, but it has to be the same for the input and output relations we are looking for. We do not consider related-keys in this article. Then in [12], rebound attacks have been used to improve known-key distinguishers on AES by Mendel et al. and in [8], Gilbert and Peyrin have used both the **SuperSBox** and the rebound techniques to get a known-key distinguisher on 8-round AES-128. Last year at FSE, Sasaki and Yasuda show in [18] an attack on 11 Feistel rounds and collision attacks in hashing mode also using rebound techniques, and more recently, Sasaki et al. studied the known-key scenario for Feistel ciphers like Camellia in [17].

Our Results. In this paper, we study 128- and 256-bit reduced versions of AES in the (single) chosen-key model where the attacker is challenged to find a key k and a pair of messages (m, m') such that $m \oplus m' \in E$ and $\texttt{AES}_k(m) \oplus \texttt{AES}_k(m') \in F$, where E and F are two known subspaces. On AES-128, we describe in that model a way to distinguish the 7-round AES in time 2^8 and the 8-round AES in time 2^{24}. In the case of the 7-round distinguisher, our technique improves the 2^{16} time complexity of a regular rebound technique [13] on the **SubBytes** layer by computing intersections of small lists. The 8-round distinguisher introduces a problem related the **SuperSBox** construction where the key parameter is under the control of the adversary. As for AES-256, the distinguishers are the natural extensions of the ones on AES-128. Our results are reported in Table 1. We have experimentally checked our results and examples are provided in the appendices. We believe our practical distinguishers can be useful to construct non-trivial inputs for the AES block cipher to be able to check the validity of some theoretical attacks, for instance [7].

Outline of the paper. The paper is organized as follows. We begin in Section 2 by recalling the AES and the concept of **SuperSBox**. Then in Section 3.1, we precise the chosen-key model in the ideal case to be able to compare our distinguishers to the ideal scenario. Section 3.1 describes the main results of the AES-128 and Section 4 shows how to apply similar results to the AES-256.

2 Description of the AES

The Advanced Encryption Standard [16] is a Substitution-Permutation Network that can be instantiated using three different key bit-lengths: 128, 192, and 256. The 128-bit plaintext

Table 1: Comparison of our results to previous ones on reduced-round distinguishers of the AES-128 in the open-key model. Results from [1] are not mentioned since we do not consider related-keys in this paper.

Target	Model	Rounds	Time	Memory	Ideal	Reference
AES-128	Known-key	7	2^{56}	-	2^{58} ⋆	[9]
	Known-key	7	2^{24}	2^{16}	2^{64}	[12]
	Single-chosen-key	7	2^{22}	-	2^{64}	[3]
	Single-chosen-key	**7**	$\mathbf{2^{8}}$	$\mathbf{2^{8}}$	$\mathbf{2^{64}}$	**Section 3.2**
	Known-key	8	2^{48}	2^{32}	2^{64}	[8]
	Single-chosen-key	8	2^{44}	-	2^{64}	[3]
	Single-chosen-key	**8**	$\mathbf{2^{24}}$	$\mathbf{2^{16}}$	$\mathbf{2^{64}}$	**Section 3.3**
AES-256	**Single-chosen-key**	**7**	$\mathbf{2^{8}}$	$\mathbf{2^{8}}$	$\mathbf{2^{64}}$	**Section 4.1**
	Single-chosen-key	**8**	$\mathbf{2^{8}}$	$\mathbf{2^{8}}$	$\mathbf{2^{64}}$	**Section 4.2**
	Single-chosen-key	**9**	$\mathbf{2^{24}}$	$\mathbf{2^{16}}$	$\mathbf{2^{64}}$	**Section 4.3**

⋆ Claimed by the authors as a *very inaccurate estimation of the [ideal] complexity.*

initializes the internal state viewed as a 4×4 matrix of bytes as values in the finite field $GF(2^8)$, which is defined via the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ over $GF(2)$. Depending on the version of the AES, N_r rounds are applied to that state: $N_r = 10$ for AES-128, $N_r = 12$ for AES-192 and $N_r = 14$ for AES-256. Each of the N_r AES round (Figure 1) applies four operations to the state matrix (except the last one where we omit the **MixColumns**):

- **AddRoundKey** (AK) adds a 128-bit subkey to the state.
- **SubBytes** (SB) applies the same 8-bit to 8-bit invertible S-Box S 16 times in parallel on each byte of the state,
- **ShiftRows** (SR) shifts the i-th row left by i positions,
- **MixColumns** (MC) replaces each of the four column C of the state by $M \times C$ where M is a constant 4×4 maximum distance separable circulant matrix over the field $GF(2^8)$, $M = \mathrm{circ}(2, 3, 1, 1)$.

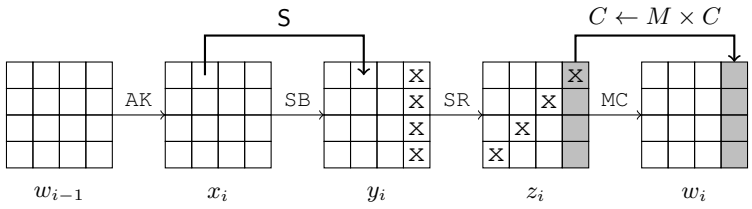
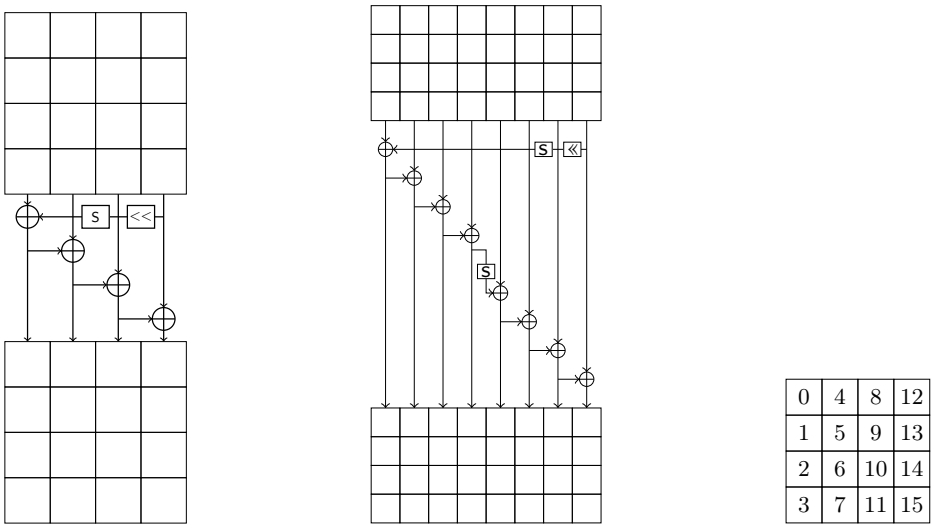


Figure 1: An AES round applies MC ∘ SR ∘ SB ∘ AK to the state. There are $N_r = 10$ rounds in AES-128.

After the N_r-th rounds has been applied, a final subkey is added to the internal state to produce the ciphertext. The key expansion algorithm to produce the $N_r + 1$ subkeys for AES-128 is described in Figure 2(a), and in Figure 2(b) for the AES-256. We refer to the official specification document [16] for further details.

SuperSBox. In [6], Rijmen and Daemen introduced the concept of **SuperSBox** to study two rounds of AES. This transformation sees the composition SB ∘ AK(k) ∘ MC ∘ SB as four parallel applications of a 32-bit S-Box, and has been useful for several cryptanalysis works, see for instance [8,10]. Abusing notations, in the sequel, we call **SuperSBox** keyed by the key k the transformation that applies this composition to a single AES-column. In that



(a) AES-128. (b) AES-256. (c) Naming.

Figure 2: Key schedules of the variants of the AES (AES-128 and AES-256) – Naming of bytes in a state.

context, the key k which parameterized the **SuperSBox** is also a 32-bit AES-column. We denote that operation by $\mathbf{SuperSBox}_k$.

Notations. In this paper, we count the AES rounds from 0 and we refer to a particular byte of an internal state x by $x[i]$, as depicted in Figure 2(c). Moreover, as shown in Figure 1, in the ith round, we denote the internal state after **AddRoundKey** by x_i, after **SubBytes** by y_i, after **ShiftRows** by z_i and after **MixColumns** by w_i. To refer to the difference in a state x, we use the notation Δx.

3 Chosen-key distinguishers

3.1 Limited Birthday Distinguishers

In this section, we precise the distinguishers we are using. Our first goal is to distinguish the AES-128 from an ideal keyed-permutation in the chosen-key model. We will derive distinguishers for AES-256 afterwards. We are interested in the kind of distinguishers where the attacker is asked to find a key and a pair of plaintext whose difference is constrained in a predefined input subspace such that the ciphertext difference lies in an other predefined subspace.

Property 1 *Given two subspaces E_{in} and E_{out}, a key k and a pair of messages (x, y) verify the property on a permutation P if $x + y \in E_{in}$ and $P(x) + P(y) \in E_{out}$.*

This type of distinguisher looks like the limited birthday distinguishers introduced by Gilbert and Peyrin in [8] with a very close lower bound proved in [15], except that we allow the attacker more freedom; namely, in the choice of the key bits. To determine how hard this problem is, we need to compare the real-world case to the ideal scenario. In the latter, the attacker faces a family[1] of pseudo-random permutations $\mathcal{F} : \mathcal{K} \times \mathcal{D} \longrightarrow \mathcal{D}$, and would run a limited birthday distinguisher on a particular random permutation F_k to find a pair of

[1] where both $\mathcal{K}$ and $\mathcal{D}$ are $\{0,1\}^{128}$ in the case of AES-128.

messages that conforms to the subspace restrictions of Property 1. The additional freedom of this setting does not help the attacker to find the actual pair of messages that verifies the required property, because the permutation F_k has to be chosen beforehand. Put it another way, the birthday paradox is as constrained as if the key were known since no difference can be introduced in the key bits.

Therefore, even if we let the key to be chosen by the attacker, the limited birthday distinguisher from [8] applies in the same way. For known E_{in} and E_{out}, we denote $n_i = \dim(E_{in})$ and $n_o = \dim(E_{out})$. In terms of truncated differences, n_i (resp. n_o) represents the number of independent active truncated differences in the input (resp. output) of a random permutation $F_k \in \mathcal{F}$ (see Figure 3). Both n_i and n_o range in the interval between 0 and n, where $n = 16$ in the case of AES. Without loss of generality, we assume that $n_i \leq n_o$: the attacker thus considers F_k rather than its inverse, as it is easier to collide on $n - n_o$ differences than on $n - n_i$. The attacker continues by constructing two lists L and L' of

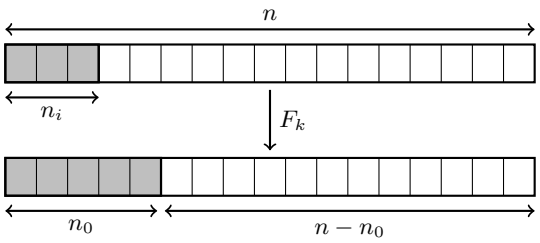


Figure 3: Assuming $n_i \leq n_o$, the attacker searches for a pair of input to the random permutation F_k differing in n_i known byte positions such that the output differs in n_o known byte positions. A gray cell indicates a byte with a truncated difference.

2^{8n_i} plaintexts each by choosing a random value for the $n - n_i$ inactive bytes of the input and considering all the n_i active ones in E_{in}. With a birthday paradox on the two lists L and L', she expects a collision on at most $2n_i$ bytes of the ciphertexts. In the event that $n - n_o \geq 2n_i$, then $n - 2n_i$ bytes have not a zero-difference in the ciphertext. Hence, we need to restart the birthday paradox process about $2^{8(n-n_o-2n_i)}$ times, which costs $2^{8(n-n_o-n_i)}$ in total. Otherwise, if $n - n_o < 2n_i$, then a single birthday paradox with lists of size $2^{8(n-n_o)/2}$ is sufficient to get a collision on the $n - n_o$ required bytes in time $2^{8(n-n_o)/2}$.

3.2 Distinguisher for 7-round AES-128

We consider the 7-round truncated differential characteristic of Figure 4, where the differences in both the plaintext and the ciphertext lie in subspaces of dimension four. Indeed, the output difference lies in a subspace of dimension four since all the operations after the last **SubBytes** layer are linear. With respect to the description of the distinguisher (Section 3.1), the time complexity to find a pair of messages that conforms to those patterns in a family of pseudo-random permutations is 2^{64} basic operations.

The following of this section describes a way to build a key and a pair of messages that conform to the restrictions in time 2^8 basic operations using a memory complexity of 2^8 bytes. This complexity has to be compared to 2^{16} operations, which is the time complexity expected for a straightforward application of the rebound attack [13] on the **SubBytes** layer of the AES. In that case, there are 16 random differential transitions around the

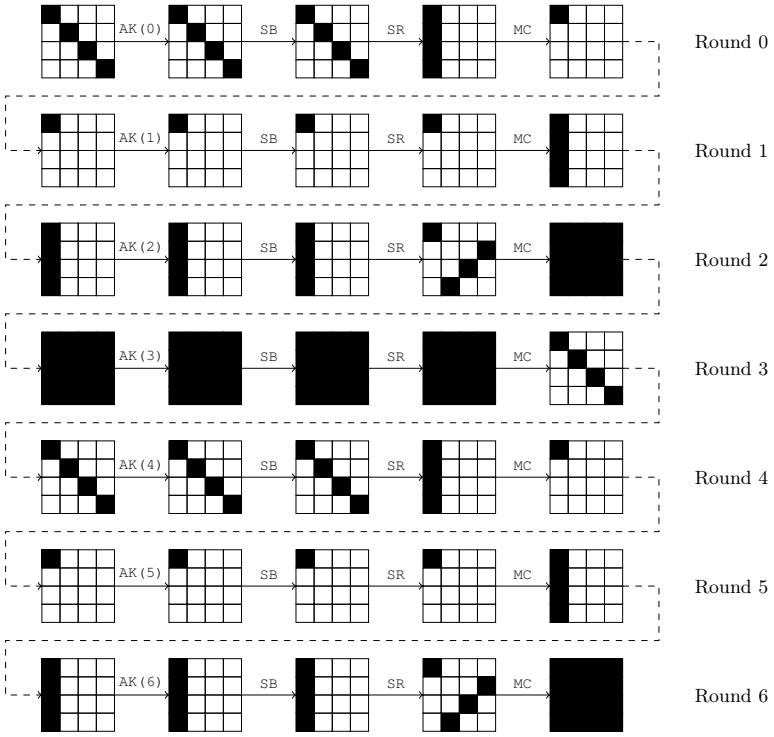


Figure 4: The 7-round truncated differential characteristic used to distinguish the AES-128 from a random permutation. Black bytes are active, white bytes are not.

AES S-Box, which happens to be all compatible[2] with probability 2^{-16}. Repeating with random differences 2^{16} times, we expect to find a pair of internal states that conforms to the randomized differences. In the following, we proceed slightly differently to reach a solution in time 2^8.

In terms of freedom degrees, we begin by estimating the number of solutions that we expect to verify the truncated differential characteristic. There are 16 bytes in the first message, 4 more independent ones in the second message and 16 others in the key: that makes 36 freedom degrees at the input. On a random input, the probability that the truncated differential characteristic being followed depends on the amount of freedom degrees that we loose in probabilistic transitions within the **MixColumns** transitions: 3 in round 0 to pass one $4 \rightarrow 1$ truncated transition, 12 in round 3 to pass four $4 \rightarrow 1$ transitions and 3 again in round 4 for the last $4 \rightarrow 1$ transition. In total, we thus expect

$$2^{8\times(16+4+16)}\, 2^{-8\times(3+12+3)} = 2^{8\times 18}$$

triplets (m, m', k) composed by a pair (m, m') of messages and a key k to conform to the truncated differential characteristic of Figure 4. Hence, we have 18 freedom degrees left to find such a triplet.

First, we observe that whenever we find such a solution for the middle rounds (round 1 to round 4), we are ensured that all the rounds will be covered as in the whole truncated differential characteristic due to an outward propagation occurring with probability 1. Hence, our strategy focuses on those rounds. The context is similar to the rebound scenario, where we first solve the inbound phase and then propagate it into the outbound phase.

[2]By compatible, we mean that we can find at least a pair of values that conforms to the differential transition. In the case of the AES S-Box, for a random differential transition $\delta \rightarrow \delta'$, this is known to be possible with probability close to 1/2.

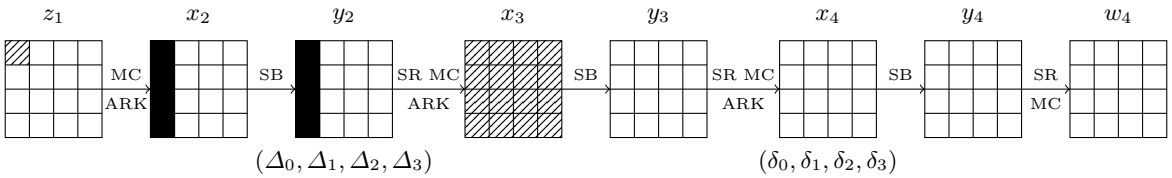


Figure 5: The 7-round distinguishing attack focuses of the middle rounds. Black bytes have known values and differences, gray bytes have known values, hatched bytes have known differences and white bytes have unknown values and/or differences.

To reduce the number of valid solutions, we begin by fixing some bytes (Figure 5) to a random value: Δz_1 and $x_2[0..3]$. Therefore, we can deduce the values and differences in the first column of x_2 and y_2, as well as the difference Δx_3 by linearity. Let $[\Delta_0, \Delta_1, \Delta_2, \Delta_3]^{\mathrm{T}}$ be the column-vector of deduced differences in Δy_2 and $\mathrm{diag}(\delta_0, \delta_1, \delta_2, \delta_3)$ the differences in the diagonal of Δx_4. Linearly, we can express the differences around the **SubBytes** layer of round 3 (see Figure 6). As a consequence, from the differential properties of the `AES` S-Box,

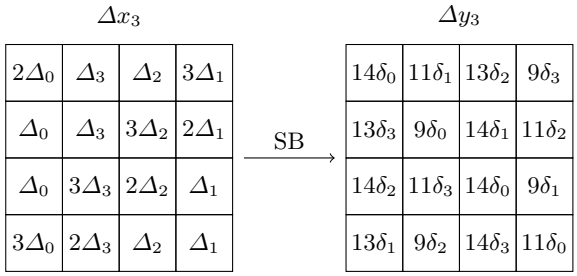


Figure 6: Differences around the **SubBytes** layer of round 3: each Δ_j is fixed, whereas the δ_i are yet to be determined.

for $i, j \in \{0, \ldots, 3\}$, Δ_j suggests 2^7 different values for δ_i: we store them in the list $L_{i,j}$.

$$L_{i,j} = \left\{ \delta_i \;\middle/\; \Delta_j \to \delta_i \text{ is possible} \right\}. \tag{1}$$

Once done, we build the list L_i, for $i \in \{0, \ldots, 3\}$:

$$L_i = \bigcap_{j=0}^{3} L_{i,j} = \left\{ \delta_i \;\middle/\; \forall j \in \{0, \ldots, 3\},\ \Delta_j \to \delta_i \text{ is possible} \right\}. \tag{2}$$

Each $L_{i,j}$ being of size 2^7, we expect each L_i to contain 2^4 elements.

We continue by setting $\Delta x_4[0]$ to random value in L_0 and $x_4[0]$ to a random value, which allow to determine the value and difference in $y_4[0]$. Since the difference Δy_4 can only take 2^8 values due to the **MixColumns** transition of round 4, we also deduce Δw_4 and the remaining differences in Δy_4. The knowledge of Δy_4 suggests 2^7 possible values for δ_i. As before, we store them in lists called T_i, and we select a value for δ_i in $L_i \cap T_i$ (Figure 7). We expect each intersection to contain about 2^3 elements. More rigorously, if we assume that the lists $L_{i,j}$ and T_i are uniformly distributed, then the probability that L_0, $L_1 \cap T_1$, $L_2 \cap T_2$ and $L_3 \cap T_3$ are not empty is higher than 99.96% (see proof in Appendix C). Finally, we compute the values in x_3 and in the diagonal of x_4.

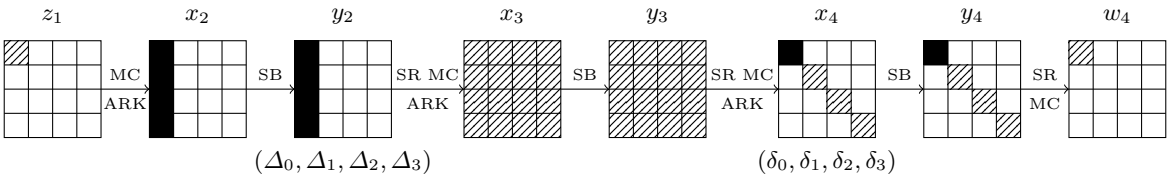


Figure 7: The 7-round distinguishing attack focuses of the middle rounds. Black bytes have known values and differences, gray bytes have known values, hatched bytes have known differences and white bytes have unknown values and/or differences.

We now need to find a key that matches the previous solving in the internal states: we build a partial pair of internal states that conforms to the middle rounds, but that sets 8 bytes on constraints in the key. Namely, if we denote k_i the subkey introduced in round i and $u_i = \mathsf{MC}^{-1}(k_i)$, then both u_3 and k_4 have four known bytes (see Figure 8). We start by

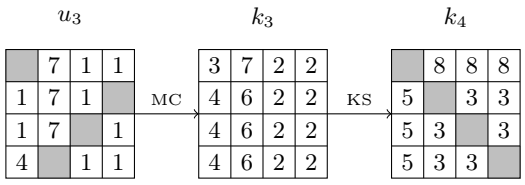


Figure 8: Generating a compatible key: gray bytes are known, and numbers indicate the order in which we guess or determine the bytes.

fixing all the bytes marked by `1` in u_3 to random values: this allows to compute the values of all `2`'s in the two last columns of k_3. By the column-wise operations of `AES` key schedule, we can get the values of all bytes marked by `3`. As for the `4`'s, we get them since there are four known bytes among the eight in the first columns of u_3 and k_3. Again, the key schedule gives the `5`'s and `6`'s, and the **MixColumns** the `7`'s. Finally, we determine values for all the byte tagged by `8` from the key schedule equations. By inverting the key schedule, we are thus able to compute the master key k.

All in all, we start by getting a partial pair of internal states that conforms to the middle rounds, continue by deriving a valid key that matches the partial known bytes and determine the rest of the middle internal states to get the pair on input messages. The bottleneck of the time and memory complexity occurs when handling the lists of size at most 2^8 elements to compute intersections. Note that those intersections can be done in roughly 2^8 operations by representing lists by 256-bit numbers and then perform a logical AND.

In the end, we build a pair of messages (m, m') and a key k that conforms to the truncated differential characteristic of Figure 4 in time 2^8 basic operations, where it costs 2^{64} in the generic scenario. We note that among the 18 freedom degrees left for the attack, we used only 10 by setting 10 bytes to random values, such that we expect $2^{8\times 8} = 2^{64}$ solutions in total. All those solutions could be generated in time 2^{64} by iterating over all the possibilities of the bytes marked by `1` in Figure 8.

We implemented the described algorithm to verify that it indeed works, and we found for instance the triplet (m, m', k) reported in Appendix A.

3.3 Distinguisher for 8-round AES-128

We consider the 8-round truncated differential characteristic of Figure 9, where the matrices of differences in both the plaintext and the ciphertext lie in the same matrix subspaces of dimension four as before. Indeed, the output difference lies in a subspace of dimension four since all the operations after the last **SubBytes** layer are linear. Again, the distinguisher previously described (Section 3.1) claims that the time complexity to find a pair of messages that conforms to those patterns in a family of pseudo-random permutations runs in time 2^{64} operations.

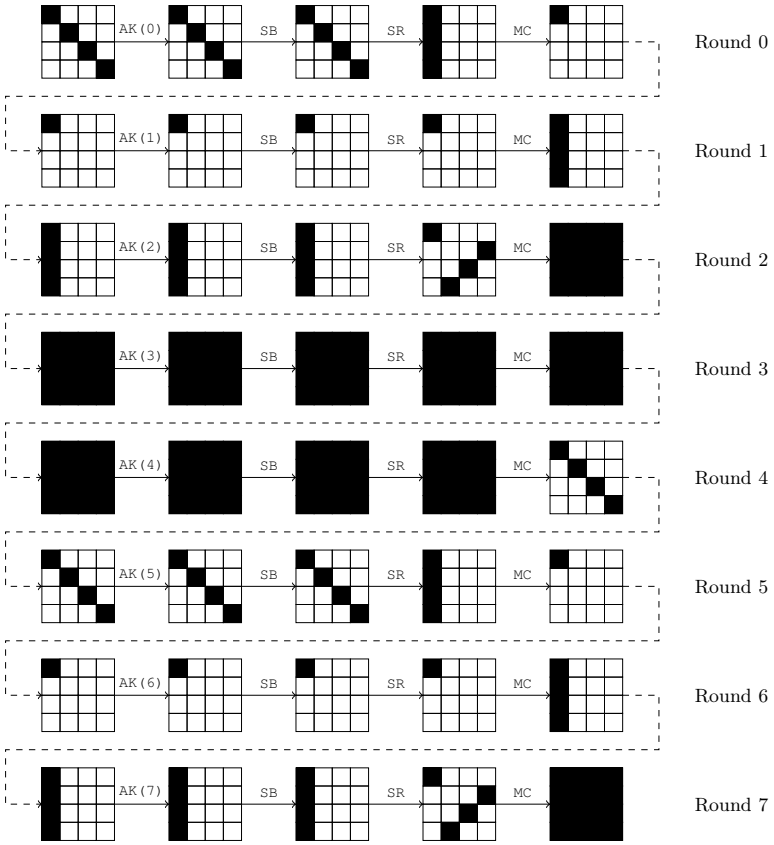


Figure 9: The 8-round truncated differential characteristic used to distinguish the AES-128. Black bytes are active, white bytes are not.

The following of this section describes a way to build a key and a pair of messages that conform to the restrictions in time and memory complexity 2^{24}. We note that it is possible to optimize the memory requirement to 2^{16}. As in the previous section, there are 36 freedom degrees at the input, which shrink to 18 after the consideration of the truncated differential characteristic. Therefore, we also expect $2^{8\times 18}$ solutions in the end.

First of all, we observe that finding 2^{24} triplets (m, m', k) composed by a key and a pair of internal states that conform to the rounds 2 to 5 is sufficient since the propagation in the outward rounds is done with probability 2^{-24} due to the **MixColumns** transition of round 1. The following analysis consequently focuses of those four middle rounds.

We now describe an instance of a problem that we use as a building block in our algorithm, which is related to the keyed **SuperSBox** construction.

Problem 1. *Let a and b two bytes. Given a 32-bit input and output differences Δ_{in} and Δ_{out} of a* $\mathbf{SuperSBox}_k$ *for a unknown k, find all the pairs of* AES*-columns (c, c') and keys k such that:*

i. $c + c' = \Delta_{in}$,
ii. $\mathbf{SuperSBox}_k(c) + \mathbf{SuperSBox}_k(c') = \Delta_{out}$,
iii. $\mathbf{SuperSBox}_k(c) = [a, b, \star, \star]^{\mathrm{T}}$.

Considering the key k known and the case where there is no restriction on the output bytes (*iii*), we would expect this problem to have one solution on average. Finding it would naively require 2^{32} computations by iterating over the 2^{32} possible inputs and check whether the output has the correct Δ_{out} known difference. The additional constraints on the two output bytes reduce the success of finding a pair (c, c') of input to 2^{-16}, but if we allow the four bytes in the key k to be chosen, then we expect 2^{16} solutions to this problem.

To find all of them in 2^{16} simple operations, we proceed as follows (Figure 10): the two output bytes a and b being known, we can deduce the values of the two associated bytes before the last **SubBytes**, $\tilde{a}$ and $\tilde{b}$ respectively. We can also deduce the difference in those bytes since the output difference is known. Then, we guess the two unset differences at the input of the last **SubBytes**: the differences then propagate completely inside the **SuperSBox**. At both **SubBytes** layers, by the differential properties of the AES S-Box, we expect to find one value on average for each of the six unset transitions. Consequently, the input and output of the **AddRoundKey** operation are known, which determines the four bytes of k. In the end, we find the 2^{16} solutions of Problem 1 in time 2^{16} operations.

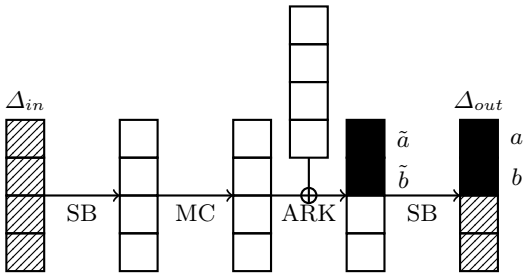


Figure 10: Black bytes have known values and differences, hatched bytes have known differences and white bytes have unknown values and/or differences.

To apply this strategy to the 8-round truncated differential characteristic of Figure 9, we start by randomizing the difference Δy_2, the difference Δw_5 and the values in the first column of w_5. Due to the linear operations involved, we deduce $\Delta x_3 = \Delta w_2$ from Δy_2 and Δy_4 from Δw_4. To use the previous algorithm, we randomize the values of the two first columns of w_4 (situtation in Figure 11). Doing so, the four columns of y_4 are constrained on two bytes each and have fixed differences. Consequently, the four **SuperSBoxes** between x_3 and y_4 keyed by the four corresponding columns of k_4 conforms to the requirements[3] of Problem 1. In time and memory complexity 2^{16}, for $i \in \{0, 1, 2, 3\}$, we store the 2^{16} solutions for the ith **SuperSBox** associated to the ith column of x_4 in the list L_i.

We continue by observing that the randomization of the bytes in w_4 actually sets the value of two diagonal bytes in k_5, $k_5[0]$ and $k_5[5]$, which imposes constraints of the elements in the lists L_i. We start by considering the 2^{16} elements of L_3, and for each of them, we

[3]The positions of the known output bytes differ, but the strategy applies in the same way.

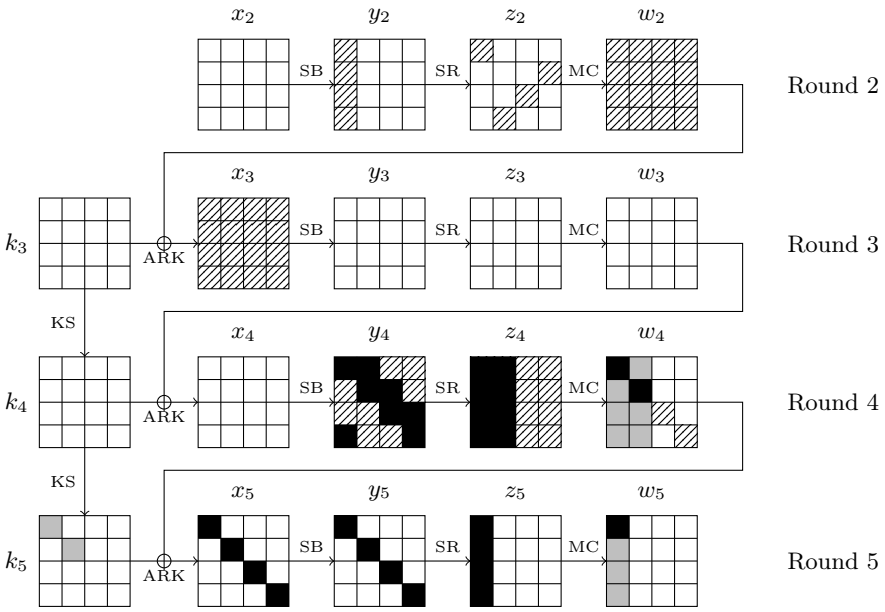


Figure 11: Black bytes have known values and differences, gray bytes have known values, hatched bytes have known differences and white bytes have unknown values and/or differences.

learn the values $x_4[12..15]$ and $k_4[12..15]$. Due to the column-wise operations in the key schedule, we also deduce the value of $k_4[0]$. Filtering the elements of L_0 which share that value of $k_4[0]$, we are left with 2^8 elements for bytes $x_4[0..3]$ and $k_4[0..3]$. At this point, we constructed $2^{16+8} = 2^{24}$ solutions in time 2^{24} that we store in a list $L_{0,3}$.

As $k_5[5]$ has been previously determined, we can deduce $k_4[5] = k_5[5] + k_5[1]$ from the AES key schedule for each of the entry of $L_{0,3}$. Again, this adds an 8-bit constraint on the elements of L_1: we expect 2^8 of them to match the condition on $k_4[5]$. In total, we could construct a list $L_{0,1,3}$ of size $2^{24+8} = 2^{32}$, whose elements would be the columns 0, 1 and 3 of x_4 and k_4, but as soon as we get 2^{24} elements in that list, we stop and discard the remaining possibilities.

Finally, to ensure the correctness of the choice in the remaining column 2, we need to consider the **MixColumns** operation in round 4 and the subkey k_5. Indeed, as soon as we choose an element in both $L_{0,1,3}$ and L_2, x_4, k_4 and k_5 become fully determined, but we need to ensure that the values $x_5[10]$ and $x_5[15]$ equal to the known ones. In particular, for $x_5[10]$, we have:

$$\begin{aligned} k_4[10] + k_5[6] &= k_5[10] && (3)\\ &= w_4[10] + x_5[10] && (4)\\ &= z_4[8] + z_4[9] + 2z_4[10] + 3z_4[11] + x_5[10], && (5) \end{aligned}$$

and for $x_5[15]$:

$$\begin{aligned} k_4[11] + k_5[7] + k_4[15] &= k_5[11] + k_4[15] && (6)\\ &= k_5[15] && (7)\\ &= w_4[15] + x_5[15] && (8)\\ &= 3z_4[12] + z_4[13] + z_4[14] + 2z_4[15] + x_5[15], && (9) \end{aligned}$$

where (3), (6) and (7) come from the key schedule, (4) and (8) from the **AddRoundKey** and (5) and (9) use the equations from the **MixColumns**. Hence, for each element of $L_{0,1,3}$,

we can compute:

$$\mathsf{S}(x_4[8]) + k_4[10] := x_5[10] + k_5[6] + \mathsf{S}(x_4[13]) + 2\,\mathsf{S}(x_4[2]) + 3\,\mathsf{S}(x_4[7]), \tag{10}$$

$$k_4[11] + 2\,\mathsf{S}(x_4[11]) := k_5[7] + k_4[15] + 3\,\mathsf{S}(x_4[12]) + \mathsf{S}(x_4[1]) + \mathsf{S}(x_4[6]) + x_5[15] \tag{11}$$

and lookup in L_2 to find $2^{16}\,2^{-8\times 2} = 1$ element that match those two byte conditions. We create the list L by adding the found element from L_2 to each entry of $L_{0,1,3}$.

All in all, in time and memory complexity 2^{24}, we build L of size 2^{24} and we now exhaust its elements to find one that passes the 2^{-24} probability of the $4 \to 1$ backward transition in the **MixColumns** of round 1. Indeed, an $a \to b$ transition in the **MixColumns** layer cancels $4 - b$ output bytes, so that it would happen with probability $2^{-8(4-b)}$ for a random input a. Consequently, we expect to find a pair (m, m') of messages and a key k that conforms to the 8-round truncated differential characteristic of Figure 9 in time 2^{24} when it requires 2^{64} computations in the ideal case.

Among the 18 available freedom degrees available to mount the attack, we uses 17 of them, which means that we expect to have 2^8 solutions. We could have them in time 2^{32}, but since we discarded 2^8 elements in the algorithm described, we get only 1 in time 2^{24}. We note that it is possible to gain a factor 2^8 in the memory requirements of our attack since we can implement the algorithm without storing the lists L_0, $L_{0,3}$ and $L_{0,1,3}$, by using hash tables for L_1, L_2 and L_3.

We also implemented the described algorithm to verify that it indeed works, and we found for instance the triplet (m, m', k) reported in Appendix B.

4 Extention to AES-256

The two distinguishers described in the previous section can be easily extended in distinguishers on the AES-256. The main idea is to use the 16 additional freedom degrees in the key to extend the truncated differential characteristics by introducing a new fully active round in the middle.

4.1 Distinguisher for 7-round AES-256

The first step of the attack described in the 7-round distinguisher on AES-128 (Section 3.2) still applies in the case of AES-256 since it does not involve the key schedule. Then, we can generate a compatible key easily since there are only two subkeys involved: we can just choose bytes of k_3 and k_4 as we want, except the imposed ones, and deduce the master key afterwards. This yields to a distinguisher with time and memory complexities around 2^8.

4.2 Distinguisher for 8-round AES-256

We use a similar approach as the 7-round distinguisher on AES-128 of Section 3.2, but the truncated differential characteristic has one more fully active round in the middle[4].

We begin by choosing values for Δz_1 and $x_2[0..3]$. This allows to deduce Δx_2, Δy_2, and Δx_3. Then, we also set random values for Δw_5 and for the diagonal of x_5 to obtain both Δx_5 and Δy_4. Now, we find a value for Δx_4, which is compatible with Δx_3 and Δy_4. Indeed, we can not take an arbitrary value for Δx_4 because the probability that it fits is very close to 2^{-32}. However, we can find a correct value with the following steps:

[4] In that case, the truncated differential characteristic is thus the one from Figure 9.

1. Store the 2^7 possible values for $\Delta x_4[0]$ in a list L_0.
2. In a similar way, make lists L_1 with $\Delta x_4[1]$, L_2 with $\Delta x_4[2]$ and L_3 with $\Delta x_4[3]$.
3. Choose a value for $(x_3[0], x_3[5], x_3[10], x_3[15])$ and compute $\Delta x_4[0..3]$.
4. If $\Delta x_4[0..3]$ is not in $L_0 \times L_1 \times L_2 \times L_3$, then go back to step 3.

On average, we go back to the step 3 only $\left(2^{8-7}\right)^4 = 2^4$ times since lists are of size 2^7. In the same way, we can obtain values for the other columns of x_4.

At this point, we computed actual values in all those internal states, and we need to generate a compatible key. Finding one can be done using the procedure described in Figure Figure 12. Bytes tagged by `1` are chosen at random, odd steps use the key schedule equations and even steps the properties of **MixColumns**.

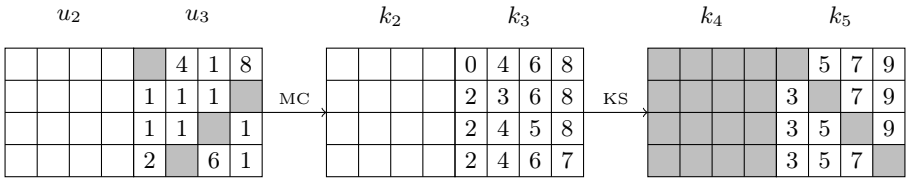


Figure 12: Generating a compatible key: gray bytes are known, and numbers indicate the order in which we guess or determine the bytes.

4.3 Distinguisher for 9-round AES-256

We begin as in Section 3.3 by choosing the difference Δy_2, the difference Δw_6 and the values in the first column of w_6. Then, we deduce $\Delta w_2 = \Delta x_3$ from Δy_2 and Δy_5 from Δw_5. In addition, we set x_3 to a random value, which allows to determine Δx_4. In order to apply the result from Problem 1 again, we set the values in two first columns of w_5 to random values.

As before, for $i \in \{0, 1, 2, 3\}$, we store in the list L_i the 2^{16} possible values of the i-th column of x_5 and the i-th column of k_5. Unlike previously, we also obtain values of the i-th column of $\mathsf{SR}(k_4)$, but the scenario of the attack still applies. We start by observing that bytes of L_0 allow to compute $k_4[1]$ and $k_4[13]$, which are bytes of L_3. Thus, we can merge L_0 and L_3 in a list $L_{0,3}$ containing 2^{16} elements. Then, we construct the list $L_{0,2,3}$ containing 2^{24} elements of $L_{0,3} \times L_2$. Finally, from bytes of $L_{0,2,3}$, we can compute:

$$3z_5[11] := k_4[2] + \mathsf{S}(k_5[15]) + k_4[6] + k_4[10] + z_5[8] + z_5[9] + 2z_5[10] + x_6[10], \quad (12)$$

$$z_5[14] + k_4[3] := \mathsf{S}(k_5[12]) + k_4[7] + k_4[11] + k_4[15] + 3z_5[12] + z_5[13] + 2z_5[15] + x_6[15]. \quad (13)$$

As a consequence, we expect only one element of L_1 to satisfy those two byte conditions and so, we obtain 2^{24} solutions for the middle rounds. All in all, this yields to a distinguisher with a time complexity around 2^{24} and a memory requirement around 2^{16} using the same trick given in Section 3.3.

5 Conclusion

In this paper, we study the Advanced Encryption Standard and show how to find a pair of messages and a key that satisfy some property a lot more efficiently than a generic attack based on the birthday paradox for both AES-128 and AES-256. Our new results improve

the previous claimed ones by reaching very practical complexities, and give new insights of the open-key model for block ciphers, and hash functions based on block ciphers.

On AES-128, we show efficient distinguishers for versions reduced to seven and eight rounds, and verified in practice that they indeed work by implementing the actual attacks. We describe precisely the algorithms to get the valid inputs, and by applying the same strategy, we deduce similar results for AES-256. Namely, we get efficient distinguishers on versions reduced to seven, eight and nine rounds.

References

1. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. [11] 1–18
2. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and Related-Key Attack on the Full AES-256. In Halevi, S., ed.: CRYPTO. Volume 5677 of Lecture Notes in Computer Science., Springer (2009) 231–249
3. Biryukov, A., Nikolic, I.: A New Security Analysis of AES-128. CRYPTO 2009 rump session, slides only (2009)
4. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In Lee, D.H., Wang, X., eds.: ASIACRYPT. Volume 7073 of Lecture Notes in Computer Science., Springer (2011) 344–371
5. Bouillaguet, C., Derbez, P., Fouque, P.A.: Automatic search of attacks on round-reduced AES and applications. In Rogaway, P., ed.: CRYPTO. Volume 6841 of Lecture Notes in Computer Science., Springer (2011) 169–187
6. Daemen, J., Rijmen, V.: Understanding Two-Round Differentials in AES. In Prisco, R.D., Yung, M., eds.: SCN. Volume 4116 of Lecture Notes in Computer Science., Springer (2006) 78–94
7. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In Abe, M., ed.: ASIACRYPT. Volume 6477 of Lecture Notes in Computer Science., Springer (2010) 158–176
8. Gilbert, H., Peyrin, T.: Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In Hong, S., Iwata, T., eds.: FSE. Volume 6147 of Lecture Notes in Computer Science., Springer (2010) 365–383
9. Knudsen, L.R., Rijmen, V.: Known-Key Distinguishers for Some Block Ciphers. In Kurosawa, K., ed.: ASIACRYPT. Volume 4833 of Lecture Notes in Computer Science., Springer (2007) 315–324
10. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound Distinguishers: Results on the Full Whirlpool Compression Function. [11] 126–143
11. Matsui, M., ed.: Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. In Matsui, M., ed.: ASIACRYPT. Volume 5912 of Lecture Notes in Computer Science., Springer (2009)
12. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher. In Jr., M.J.J., Rijmen, V., Safavi-Naini, R., eds.: Selected Areas in Cryptography. Volume 5867 of Lecture Notes in Computer Science., Springer (2009) 16–35
13. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In Dunkelman, O., ed.: FSE. Volume 5665 of Lecture Notes in Computer Science., Springer (2009) 260–276
14. Minier, M., Phan, R.C.W., Pousse, B.: Distinguishers for Ciphers and Known Key Attack against Rijndael with Large Blocks. In Preneel, B., ed.: AFRICACRYPT. Volume 5580 of Lecture Notes in Computer Science., Springer (2009) 60–76
15. Nikolic, I., Pieprzyk, J., Sokolowski, P., Steinfeld, R.: Known and Chosen Key Differential Distinguishers for Block Ciphers. In Rhee, K.H., Nyang, D., eds.: ICISC. Volume 6829 of Lecture Notes in Computer Science., Springer (2010) 29–48
16. NIST: Advanced Encryption Standard (AES), FIPS 197. Technical report, NIST (November 2001)
17. Sasaki, Y., Emami, S., Hong, D., Kumar, A.: Improved known-key distinguishers on feistel-sp ciphers and application to camellia. In Susilo, W., Mu, Y., Seberry, J., eds.: ACISP. Volume 7372 of Lecture Notes in Computer Science., Springer (2012) 87–100
18. Sasaki, Y., Yasuda, K.: Known-Key Distinguishers on 11-Round Feistel and Collision Attacks on Its Hashing Modes. In Joux, A., ed.: FSE. Volume 6733 of Lecture Notes in Computer Science., Springer (2011) 397–415

A Solution for the 7-round truncated differential characteristic on AES-128

Table 2: Example of a pair of messages (m, m') that conforms to the 7-rounds truncated differential characteristic for AES-128 of Section 3.2. The master key found by the attack is: 93CA1344 10A7EBDF B659C8AF ECC59699. The lines in this array contains the values of two internal states before entering the corresponding round, as well as their difference.

Round	m	m'	$m \oplus m'$
Init.	E5FC5DFE 79A851F7 7EB9E366 51C3D9C5	F8FC5DFE 79C951F7 7EB96566 51C3D96E	1D000000 00610000 00008600 000000AB
0	76364EBA 690FBA28 C8E02BC9 BD064F5C	6B364EBA 696EBA28 C8E0ADC9 BD064FF7	1D000000 00610000 00008600 000000AB
1	65CC94D1 85BE1AD3 F3D75BF1 ACCBB8BD	8DCC94D1 85BE1AD3 F3D75BF1 ACCBB8BD	E8000000 00000000 00000000 00000000
2	E93319CD 88F41390 10623230 F66BFBAD	C92309FD 88F41390 10623230 F66BFBAD	20101030 00000000 00000000 00000000
3	89C79074 E09E6F44 F1DBAB2F F984FCC4	1404532A 09774F8D 24BF1AFA CD551921	9DC3C35E E9E920C9 D564B1D5 34D1E5E5
4	867A12E6 BF19139C 1C848362 400030D3	047A12E6 BF5B139C 1C847C62 400030D7	82000000 00420000 0000FF00 00000004
5	84606BEA 0E22D904 3BF29061 9F454807	4B606BEA 0E22D904 3BF29061 9F454807	CF000000 00000000 00000000 00000000
6	FF867544 274436AF 75ECC287 A6BF72F6	3C6A996B 274436AF 75ECC287 A6BF72F6	C3ECEC2F 00000000 00000000 00000000
End	C49E4CB3 0C944043 D5ED6D3B 247E3843	2563B1AF 68F0EC8B A6788B48 EEF27E05	E1FDFD1C 6464ACC8 7395E673 CA8C4646

B Solution for the 8-round truncated differential characteristic on AES-128

Table 3: Example of a pair of messages (m, m') that conforms to the 8-round truncated differential characteristic for AES-128 of Section 3.3. The master key found by the attack is: 98C45623 6CA00686 301E836D 614DFAB0. The lines in this array contains the values of two internal states before entering the corresponding round, as well as their difference.

Round	m	m'	$m \oplus m'$
Init.	9588B342 D43D04D4 AB298AE1 E43687DB	0B88B342 D46904D4 AB29D0E1 E4368728	9E000000 00540000 00005A00 000000F3
0	0D4CE561 B89D0252 9B37098C 857B7D6B	934CE561 B8C90252 9B37538C 857B7D98	9E000000 00540000 00005A00 000000F3
1	53FEBB0F 6BFF8E5E B471A8E3 1A2232A3	0EFEBB0F 6BFF8E5E B471A8E3 1A2232A3	5D000000 00000000 00000000 00000000
2	E9F44380 991A8ECB F7B18344 2C936CEB	65B2054A 991A8ECB F7B18344 2C936CEB	8C4646CA 00000000 00000000 00000000
3	2977F65C 3883EDEF 615D3C9E 5CE5384B	8F24A5A9 2398C0D9 10CEDEEF DFEEB0C3	A65353F5 1B1B2D36 7193E271 830B8888
4	BB1DB144 2BE947C3 5FCD89DF DF1CA0EB	82188658 42FFCAAE B337F0CA 09AB1513	3905371C 69168D6D ECFA7915 D6B7B5F8
5	C3E1961D 02A9713E 770A20D4 5470FA8F	8DE1961D 029B713E 770A3AD4 5470FA27	4E000000 00320000 00001A00 000000A8
6	D79D534C 33CC3861 76635DCD 548870C9	EB9D534C 33CC3861 76635DCD 548870C9	3C000000 00000000 00000000 00000000
7	D7F645C6 89358035 09847940 D831EFDE	0211A2F4 89358035 09847940 D831EFDE	D5E7E732 00000000 00000000 00000000
End	16E58308 DFD78F11 A8B05B9D C0A0363E	E49CFA83 D4DC9207 FC4CF3C9 9B3BF6FE	F279798B 0B0B1D16 54FCA854 5B9BC0C0

C Probability of success

We are interested in the probability that the intersection of four or five subsets of $\{1, \ldots, 255\}$ each of size 128 being empty.

To evaluate it, let $\mathcal{P}$ denote the set of subsets $X \subset \{1, \ldots, 255\}$ such that $|X| = 128$. We also define:

$$T(n,k) := \{(X_1, \ldots, X_n) \in \mathcal{P}^n \mid |X_1 \cap \ldots \cap X_n| = k\} \qquad \text{for } n \geq 1, k \geq 0.$$

In others words, $|T(n,k)|/|\mathcal{P}^n|$ is the probability that the intersection of n elements from $\mathcal{P}$ has a size equal to k.

Property 2 *The cardinality of $T(n,k)$ satisfies the following recurrence relation:*

$$\begin{cases} |T(1,k)| = |\mathcal{P}| \text{ if } k = 128, 0 \text{ otherwise} \\ |T(n+1,k)| = \sum_{l=k}^{128} |T(n,l)| \binom{l}{k} \binom{255-l}{128-k} \end{cases} \qquad \text{for } n \geq 1, k \geq 0.$$

Proof. First, we note that we can partition $\mathcal{P}^n$ by the sets:

$$T(n,Y) := \{(X_1, \ldots, X_n) \in \mathcal{P}^n \mid X_1 \cap \ldots \cap X_n = Y\} \qquad \text{for any subset } Y \subset \{1, \ldots, 255\}.$$

Then, we have:

$$\begin{aligned} |T(n+1,k)| &= \sum_Y |\{(X_1, \ldots, X_{n+1}) \in T(n,Y) \times \mathcal{P} \mid |Y \cap X_{n+1}| = k\}| \\ &= \sum_Y |T(n,Y)| \times |\{X \in \mathcal{P} \mid |Y \cap X| = k\}| \end{aligned}$$

If we fix a set $Y \subset \{1, \ldots, 255\}$, then a set $X \in \mathcal{P}$ such that $|X \cap Y| = k$ is obtained by choosing k elements in Y and $128 - k$ elements in Y^c. As a consequence, we obtain:

$$\begin{aligned} |T(n+1,k)| &= \sum_Y |T(n,Y)| \binom{|Y|}{k} \binom{255-|Y|}{128-k} \\ &= \sum_{l=0}^{255} \binom{l}{k} \binom{255-l}{128-k} \sum_{|Y|=l} |T(n,Y)| \end{aligned}$$

Finally, we remark that $\{T(n,Y)\}_{|Y|=l}$ is a partition of $T(n,l)$ and thus:

$$|T(n+1,k)| = \sum_{l=0}^{255} \binom{l}{k} \binom{255-l}{128-k} |T(n,l)|.$$

□

Using Maple, we found that the probability of failure of the distinguisher described in Section 3.2 is:

$$\frac{T(4,0)}{|\mathcal{P}|^4} \times \left(\frac{T(5,0)}{|\mathcal{P}|^5}\right)^3 \approx 0.04\%.$$